FERMILAB-CONF-21-068-SCD

Apprentice for Event Generator Tuning

Mohan Krishnamoorthy[1,*], *Holger* Schulz[2,**], *Xiangyang* Ju[4], *Wenjing* Wang[4], *Sven* Leyffer[1], *Zachary* Marshall[4], *Stephen* Mrenna[3], *Juliane* Müller[4], and *James B.* Kowalkowski[3]

[1]Argonne National Laboratory, Lemont, IL 60439
[2]Department of Computer Science, Durham University, South Road, Durham DH1 3LE, UK
[3]Fermi National Accelerator Laboratory, Batavia, IL 60510
[4]Lawrence Berkeley National Laboratory, Berkeley, CA 94720

Abstract. APPRENTICE is a tool developed for event generator tuning. It contains a range of conceptual improvements and extensions over the tuning tool PROFESSOR. Its core functionality remains the construction of a multivariate analytic surrogate model to computationally expensive Monte-Carlo event generator predictions. The surrogate model is used for numerical optimization in chi-square minimization and likelihood evaluation. *Apprentice* also introduces algorithms to automate the selection of observable weights to minimize the effect of mis-modeling in the event generators. We illustrate our improvements for the task of MC-generator tuning and limit setting.

1 Introduction

Monte Carlo-based (MC) event generators are necessary tools for interpreting data at the high energy frontier. MCs contain O(10-100) parameters that are tuned to match selected data to allow predictions for other sets of data. In many cases, the limiting factor to precision physics at the LHC is our lack of confidence in the MC predictions. The tuning task is daunting because of the large number of parameters that should be explored and the computational cost of simulations. For the large datasets of collider data that are available for MC tuning, a common heuristic to evaluate the goodness of a prediction is:

$$\chi^2(\mathbf{p}, \mathbf{w}) = \sum_{O \in \mathcal{S}_O} w_O \sum_{b \in O} \frac{(t_b(\mathbf{p}) - \mathcal{R}_b)^2}{\Delta t_b(\mathbf{p})^2 + \Delta \mathcal{R}_b^2}, \tag{1}$$

where $\mathcal{S}_O$ is the set of observables O used in the tune, each observable has a weight w_O represented by a vector $\mathbf{w}$, $t_b(\mathbf{p})/\mathcal{R}_b$ is the theory prediction/reference data in a given bin b of an observable and the $\Delta t_b/\Delta \mathcal{R}_b$'s are error estimates on these quantities.

Our problem is to minimize $\chi^2(\mathbf{p}, \mathbf{w})$ as a function of the adjustable parameters $\mathbf{p}$ and possibly the observable weights $\mathbf{w}$. To accomplish this, one needs a range of theory predictions t_b for different possible parameter choices $\mathbf{p}$ and a method or principle for choosing $\mathbf{w}$. As an additional output, it is desirable to have an estimate of what range of parameters $\mathbf{p}$ are compatible with the data at a given confidence level.

In the following, we report on APPRENTICE, a successor to the tool PROFESSOR [1] that was developed to accomplish these goals.

[*]e-mail: mkrishnamoorthy@anl.gov
[**]e-mail: holger.schulz@durham.ac.uk

2 Problems Solved using APPRENTICE

PROFESSOR is a public tool that was introduced to automate and accelerate the tuning of event generators. Based on a modest number of MC simulations, it constructs a polynomial approximation surrogate to these predictions in each bin of a histogram representing the observables that drive the tuning. Given the surrogate function, an "optimal" set of tuned parameters $\mathbf{p}$ is determined by minimizing the heuristic (1).

PROFESSOR has been used successfully to create a number of MC tunes and in other contexts. Nonetheless, there are certain aspects that could be improved. First, the polynomial fit to the MC predictions is adequate for well-behaved distributions, but does not perform well for a prediction x with distribution $\sim 1/x$. Second, the suite of minimizers available is limited to those in SciPy and the CERN Minuit package. The dependence on minimizer, if any, needs to be explored. Third, the choice of weights w_O is done manually, and any optimization over them must be done in a costly, iterative process. An algorithm for automatically selecting these weights is desirable.

The desire to address these three issues has led to the development of a new public tool that is sufficiently different from its predecessor to have its own name: APPRENTICE.

In the following, we motivate the use of APPRENTICE by listing the problems that APPRENTICE can currently solve and by highlighting the advantages of using APPRENTICE to solve these problems, especially for HEP applications.

2.1 Rational Approximation as a Surrogate Function

Polynomial models are relatively easy to build and use. However, they have poor extrapolation behavior and are severely limited in their ability to cope with singularities. These drawbacks can reduce their effectiveness at representing physics models. On the other hand, rational functions (quotients of polynomials) can be considerably more effective [2, 3] at representing models that have real or apparent pole structure. Unfortunately, rational approximations can be numerically fragile to compute and are prone to having spurious singularities.

APPRENTICE is capable of computing multivariate rational approximations $r(x) = p(x)/q(x)$ to simulation data using three approaches. The first is based on the univariate methods of [4, 5] and provides a robust and efficient way to compute the coefficients of $p(x)$ and $q(x)$. Although it tries to reduce the appearance of unwanted singularities by using ideas from linear algebra to minimize the degree of $q(x)$, it does not *guarantee* that $r(x)$ will be pole free in the parameter domain D.

The second uses a constrained optimization formulation that includes structural constraints on $r(x)$ to enforce the absence of poles in D. Although this approach is computationally more expensive than the first, it guarantees that the computed approximation is free of poles for box-shaped parameter domains, which can be crucial when computing surrogate models for use in optimization. In particular, the guaranteed absence of poles ensures that subsequent optimization problems involving our rational approximations are well-defined.

The third approach is a specialization of the second that is used to efficiently find a pole-free rational approximation by making $q(x) = \mathbf{A}^T\mathbf{x}+\mathbf{b}$ linear. This is achieved using auxiliary variables to write the dual of the linear program $\min_{\mathbf{x}}\ \mathbf{A}^T\ \mathbf{x}\ +\ \mathbf{b}\ s.t.\ \mathbf{x}\ \in\ [\mathbf{L},\mathbf{U}]$ as a constraint to force the denominator to be greater than 0 or be pole-free. The advantage of this approach is that it requires solving the optimization problem only to find a pole-free rational approximation (as opposed to an iterative approach) and hence it is more efficient.

2.2 Tuning Problem

The goal of the tuning problem is to find an optimal set of physics parameters, $\mathbf{p}^*$, that minimizes the function (1). When the theory prediction $t_b(\mathbf{p})$ is based on simulated data from the MC event generator, the computational cost is high (the generation of 1 million events for a given set of parameters consumes about 800 CPU minutes on a typical computing cluster), severely limiting the number of parameter choices $\mathbf{p}$ that can be explored in the tuning. To overcome this issue, we construct a parametrization of a modest number of MC simulations i.e., of $\mathrm{MC}_b(\mathbf{p})$ and $\Delta\mathrm{MC}_b(\mathbf{p})$ of each bin b as a polynomial or rational approximation using apprentice as discussed in the previous section. Thus $t_b(\mathbf{p})$ becomes our analytic surrogate $f_b(\mathbf{p})$, which is easy to compute (it can be evaluated in milliseconds) and minimize. For performing this optimization, the solver options of Truncated Newton method [6], Newton-conjugate gradient method [7], LBFGS-B algorithm [8], and Trust region algorithm [9] from python's SciPy package are provided within apprentice.

2.3 Automatic Selection of Observable Weights

To tune parameters as discussed in the previous section, the weights w_O in $\chi^2(\mathbf{p}, \mathbf{w})$ need to be specified. In practice, the weights are adjusted manually, based on experience and physics intuition: the expert fixes the weights and minimizes (1) over the parameters $\mathbf{p}$. If the resulting fit is unsatisfactory, a new set of weights is selected, and the optimization over $\mathbf{p}$ is repeated until the tuner is satisfied. Our goal is to automate the weight adjustment, yielding a less subjective and less time-consuming process.

In apprentice, the automatic selection of weights is achieved using two mathematical formulations: bilevel and robust optimization. In bilevel optimization, a merit function is minimized over the set of weights. apprentice provides three options for merit functions. The first is the portfolio objective function, which is motivated by portfolio optimization in finance, where the goal is to maximize the expected return while minimizing the risk. Translated to our problem, we want to minimize the expected error over all observables while also minimizing the variance over these errors. The second and the third merit functions are based on the central values for scoring schemes with a scoring rule, $S(P, x) = -\left(\frac{x-\mu_P}{\sigma_P}\right)^2 - \log \sigma_P^2$, where model P has mean performance μ_P and variance σ_P^2. For our application, x corresponds to the simulation prediction $f_b(\mathbf{p})$, μ_P to our observation data $\mathcal{R}_b$, and the variance σ_P^2 to our data uncertainty $\Delta\mathcal{R}_b$. The second merit function maximizes the sum over all observables of the mean across all bins in that observable of each bin's scoring function. The third merit function maximizes over the median. The scoring function for bin b is defined as $\left\{\left(\frac{f_b(\widehat{\mathbf{p}}_\mathbf{w}) - \mathcal{R}_b}{\Delta\mathcal{R}_b}\right)^2 + \log(\Delta\mathcal{R}_b^2)\right\}$. Since we don't have the full analytical expression of the outer objective function, this function is approximated with a radial basis function (RBF) [10] to perform the bilevel optimization in apprentice.

Robust optimization is a single-level formulation for finding the optimal weights for $\chi^2(\mathbf{p}, \mathbf{w})$. It estimates the parameters $\mathbf{p}$ that minimize the largest deviation $(f_b(\mathbf{p}) - \mathcal{R}_b)^2$ over all bins in an uncertainty set $\mathcal{U}_b$ of bin b. Assuming that the experiment and the MC simulation are described using independent random variables with mean $\mathcal{R}_b$, the uncertainty set $\mathcal{U}_b$ for each bin b is described by the interval $[\mathcal{R}_b - \Delta\mathcal{R}_b - \Delta f_b(\mathbf{p}), \mathcal{R}_b + \Delta\mathcal{R}_b + \Delta f_b(\mathbf{p})]$. Since $(f_b(\mathbf{p}) - \mathcal{R}_b)^2$ is a square function, the largest deviation occurs at either end of the interval. Using this, the robust optimization formulation can be easily rewritten as a one shot single level optimization problem. To avoid the trivial solution of all weights being zero, the sum of the weights $\mathbf{w} \in [0, 1]$ across all the observables is required to cover at least $\mu\%$ of the total observables in $\mathcal{S}_O$, where μ is a hyperparameter set by the end user.

3 Using APPRENTICE

The source code for APPRENTICE is available at https://github.com/HEPonHPC/apprentice. For each problem described in Section 2, we provide a convenient script that the end user can use directly with appropriate input arguments to solve that problem for their use case. In this section, we first describe how to clone and install APPRENTICE, and, then, how the interface scripts can be used by the end user to directly solve their problem. For this, we describe the important arguments of each script in this section. For the complete usage, run the script using the `-h` flag. Then, we describe what output to expect from each of these scripts. Additionally, we also note the software features that are currently implemented for each problem.

3.1 Setting up APPRENTICE

To clone and install apprentice, the following commands can be used from a Unix Terminal:

```
$git clone --recurse-submodules git@github.com:HEPonHPC/apprentice.git
$cd apprentice
$pip install .
$cd pyoo
$pip install .
```

3.2 Creating Polynomial and Rational Approximations

We provide an easy-to-use script at `apprentice/bin/app-build` to create a rational approximation $r(x) = p(x)/q(x)$ for an arbitrary degree for $p(x)$ and $q(x)$ or when the order of $q(x)$ is one or zero i.e., polynomial approximation. The important arguments of the script are: (a) *args[0]*: Location of directory of YODA files or of the HDF5 file with the MC generator central values and uncertainty values for all bins to build approximations for, (b) *–order*: order of numerator polynomial and denominator polynomial separated by a comma e.g., *2,1*, (c) *–errs*: Boolean, which if True then the approximation $\Delta f_b(\mathbf{p})$ is created for MC generator uncertainty values, otherwise the approximation $f_b(\mathbf{p})$ is created for MC generator central values, (d) *–pnames*: data file with parameter names (one name per line of the text file), (e) *-t*: Number of multistarts to use for pole detection, and (f) *-o*: Output JSON file location.

The output is a JSON object, where the keys at the first level are the bin names, and, at the second level, include: (a) *dim*: Number of parameter dimensions, (b) *m*: order of numerator polynomial, (c) *n*: order of denominator polynomial (present if order of denominator polynomial is greater than 0), (e) *pcoeff*: numerator coefficients[1], and (f) *qcoeff*: denominator coefficients (present if order of denominator polynomial is greater than 0).[1]

3.3 Solving the χ^2 Tuning Problem

For solving the tuning problem described in Section 2.2, the interface script is located at `apprentice/bin/app-tune2`. The important arguments of the script are: (a) *args[0]*: Weights text file location with two columns (separated by a space) where the first column are the observable names $\mathcal{O} \in \mathcal{S}_{\mathcal{O}}$ and the second column contain $w_{\mathcal{O}}$. (b) *args[1]*: Measurement data central and uncertainty values in JSON format where the first level of keys are the bin names and the corresponding value is a two-element array containing central and uncertainty values, (c) *args[2]*: Approximation JSON file location created using the script described in Section 3.2 for the MC generator central values, (d) *-e*: Approximation JSON file location

[1] The order of the coefficients can be found in people.sc.fsu.edu/~jburkardt/py_src/monomial/monomial.html

created using the script described in Section 3.2 for the MC generator uncertainty values, (e) *-s*|*–survey*: Size of survey when determining the starting point, (f) *-r*|*–restart*: Number of restarts to use with different starting points with the aim of finding multiple local minima, (g) *–msp*: Manual start parameter vector to use (values separated by a comma), and (h) *-a*|*–algorithm*: The minimization algorithm, options include: *tnc* for Truncated Newton method, *ncg* for Newton-conjugate gradient method, *lbfgsb* for LBFGS-B algorithm, and *trust* for Trust region algorithm.

The output of this script are the parameters $\mathbf{p}^*, \chi^2(\mathbf{p}^*, \mathbf{w})$, and some other pertinent information about the optimization for logging purposes. This script also supports running the optimization with multiple starting points in parallel using mpi4py [11].

3.4 Event Generator Tuning by Automatic Selection of Observable Weights

In this section, we describe the interface scripts for solving generator tuning problems with automatic selection of observable weights.

3.4.1 Bilevel Optimization Formulation

Solving the generator tuning problem with selection of observable weights using the bilevel optimization formulation as described in Section 2.3 requires two steps: (a) generating the data to train the RBF that will be used as an approximation of the outer level objective function and (b) solving the bilevel optimization problem with either of the three merit functions of the portfolio function, the mean of the scoring function (meanscore), or the median of the scoring function (medianscore). The interface script for generating the training data is at `apprentice/pyoo/bin/pyoo-train`. The arguments *args[0]*, *args[1]*, *args[2]*, *-e*, *-s*|*–survey*, and *-r*|*–restart* of this script mean the same as those described in Section 3.3. Additional (important) arguments of the script are: (a) *-d*|*–design* Size of the initial design, (b) *-o*|*–output* Training data output file, and (c) *–seed* Random seed to use.

Using the training output data (a JSON file), we can run the interface scripts for solving the bilevel optimization using the portfolio objective found at `apprentice/pyoo/bin/pyoo-run-portfolio`, meanscore objective found at `apprentice/pyoo/bin/pyoo-run-meanscore`, and medianscore objective found at `apprentice/pyoo/bin/pyoo-run-medianscore`. The arguments for all three scripts are the same. Also, the arguments *args[0]*, *args[1]*, *args[2]*, *-e*, and *-s*|*–survey* of these scripts mean the same as those described in Section 3.3. Additional (important) arguments of the script are: (a) *args[3]*: Training data JSON file, (b) *-n*|*–niterations*: Number of iterations to use, and (c) *-o*|*–output*: Output JSON file.

The output of each script is a JSON object with keys such as *X_outer*, *X_inner*, *Y_outer*, *hnames* and *pnames*. The keys *X_outer* and *X_inner* point to an array of array where each array element in the outer array are the weights $\mathbf{w}^*$ of the observables and the parameters $\mathbf{p}^*$ obtained in each iteration of the outer optimization. The best weight and parameter element in the output corresponds to the index of the smallest value in the array against the key *Y_outer* i.e., smallest outer objective value. The order of the observables and parameter dimensions that the weights and parameter values correspond to can be found against the keys *hnames* and *pnames*, respectively.

3.4.2 Single Level Robust Optimization Formulation

To solve the generator tuning problem where the observable weights are selected using the robust optimization approach (see Section 2.3), the interface script is located at `apprentice/pyoo/bin/pyoo-robopt`. The important arguments of the script are: (a) *-w|–weights*: Weight file (see description of *arg[0]* in Section 3.3), (b) *-d|–expdata*: Measurement data (see description of *arg[1]* in Section 3.3), (c) *-a|–approx*: MC central values approximation JSON file (see description of *arg[2]* in Section 3.3), (d) *-e|–error*: See description of *-e|–error* in Section 3.3, (e) *-o|–outtdir*: Output directory where the output JSON file and the optimization log files are stored (more details below), (f) *-s|–solver*: Solver where the options are *ipopt* that makes use to IPOPT solver with AMPL solver interface (ASL) using PYOMO [12, 13] and cyipopt [14] uses the python wrapper around IPOPT and avoids making use of PYOMO and the ASL, and (g) *-m|–mu*: Hyperparameter $\mu \in [0, 1]$ is the value to be used in the constraint to avoid tivial solutions of all weights being zero.

The output of the script is a JSON object where the weights $\mathbf{w}^*$ of the observables are in the array pointed by the key *X_outer*, the parameters $\mathbf{p}^*$, which is obtained by solving the χ^2 tuning problem using $\mathbf{w}^*$, are in the array pointed by the key *X_inner*. The order of the observables and parameter dimensions that the weights and parameter values correspond to can be found against the keys *hnames* and *pnames*, respectively. Additionally, the object pointed by the key *log* contains some pertinent information about the optimization for logging purposes.

4 Impact of APPRENTICE for HEP applications

In this section, we discuss the impact of the problems that can be solved using APPRENTICE for HEP applications.

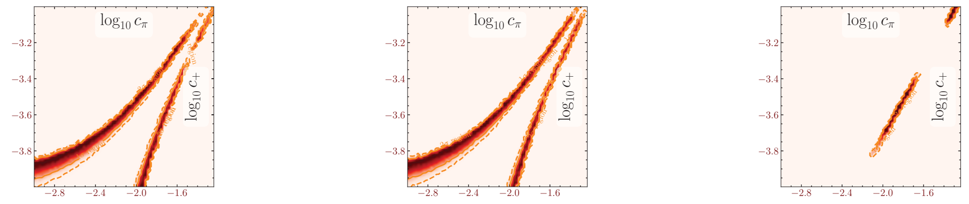
4.1 Rational Approximation for High Energy Physics

As a demonstration of the utility of rational approximations, we consider the problem of setting limits on dark matter particles in a future direct detection Xenon-based experiment (see [15, Sec. 26]). The physics model has three parameters, $x = (m_\chi, c_+, c_\pi)$, representing the dark matter particle mass (ranging over 10-100GeV/c^2) and two (dimensionless) couplings to ordinary matter (ranging from 10^{-4}-10^{-3} and 10^{-3}-10^{-1}, respectively). We consider a “binned likelihood” analysis [16] of $\mathcal{L}(x^{(k)}|d_1, d_2, \ldots, d_6) = \prod_{b=1}^{6} \frac{N_b(x^{(k)}) d_b e^{N_b(x^{(k)})}}{\Gamma(d_b + 1)}$, where the d_b represent six data points and $N_b(x^{(k)})$ denote the simulated quantities for a point $x^{(k)}$ that correspond to the d_b. Since no measured data exists as yet, we assume a signal consistent with a dark matter mass $m_\chi = 10$ GeV/c^2 and an interaction strength large enough to produce approximately 100 events, yielding $\{d_1, d_2, \ldots, d_6\} = \{70.4, 26.7, 9.8, 3.4, 1.0, 0.2\}$, simulated by using a specific framework of the generalized spin-independent response to dark matter in direct detection experiments [17, 18]. Regions of parameter space consistent with the simulated data are found using MultiNest [19–21]. This requires the evaluation of the likelihood function at tens of thousands[2] of $x^{(k)}$ to succeed, and the computational cost can be substantial. To reduce the cost, we replace the expensive simulation of N_b with the cheaper evaluation of a rational approximation surrogate. The results presented here are for rational approximations of degree $M = 4, N = 4$ as well as polynomial approximations of degree 7.[3] The maximization of the likelihood function requires about 30,000 evaluations in all cases, but is about a factor of 50 times faster using the rational or polynomial approximation.

[2]The dimension of the problem and the convergence criteria of the MultiNest algorithm strongly influence the number of required function calls.

[3]The degree is chosen such that the number of coefficients is comparable to the number of coefficients used in the rational approximations.

Figure (1) shows two-dimensional profile-likelihood projections of the three-dimensional likelihood limits when using the expensive full simulation, our rational approximation results, and a polynomial approximation. Clearly, the rational approximation faithfully represents the full simulation, while the polynomial approximation is only valid over a limited range.



(a) Full simulation using MC generator

(b) Pole-free rational approximation

(c) Polynomial approximation

Figure 1: Two-dimensional profile-likelihood projections of a 3-dimensional parameter space with superplot [22]. Regions of higher likelihood are darker. The data are normalized to the maximum observed likelihood.

4.2 Tuning HEP Event Generators by Automatic Selection of Observable Weights

As described earlier, we have developed several algorithms for adjusting the weights used in our tuning heuristic. For comparison, we re-consider a tuning exercise performed by the ATLAS collaboration [23, 24] using a large set of LHC data.

Figure 2 shows cumulative distribution of χ^2 values per bin for several different classes of observables. The results of the ATLAS expert tune and a much simpler approach using equal weights for all observables is also shown. Note that the parameters obtained from the robust optimization perform well for *Jet shapes* and *Track-jet UE*. We also see that near the variance boundary, the parameters obtained from the *Expert* tune perform better for *Multijets* and *$t\bar{t}$ gap* whereas the parameters obtained from the other approaches perform better for *Substructure*. These plots demonstrate the trade-off in fitting among the different approaches, which enables the physicist to use these results as guidance for selecting the most appropriate tuning method depending on the categories that are of greater significance.

5 Conclusion

In this paper, we propose an improved software package called APPRENTICE that enables the construction of pole-free rational approximation, allows for solving the χ^2 minimization problem using a various optimization algorithms that can detect multiple local minima, and provides multiple optimization formulations to automatically assign weights to the observables yielding a less subjective and less time-consuming process to find the optimal event generator tune. In the future, we are interested in (a) tuning event generators using the MC generator data directly using derivative free optimization methods, (b) providing non-linear optimization algorithms to minimize χ^2 that are robust against all local minima, and (c) exploring machine learning techniques for dimensionality reduction and for improving the automatic weight assignment toward obtaining optimal generator tune. If your are interested in these problems or in any other interesting extensions to APPRENTICE,

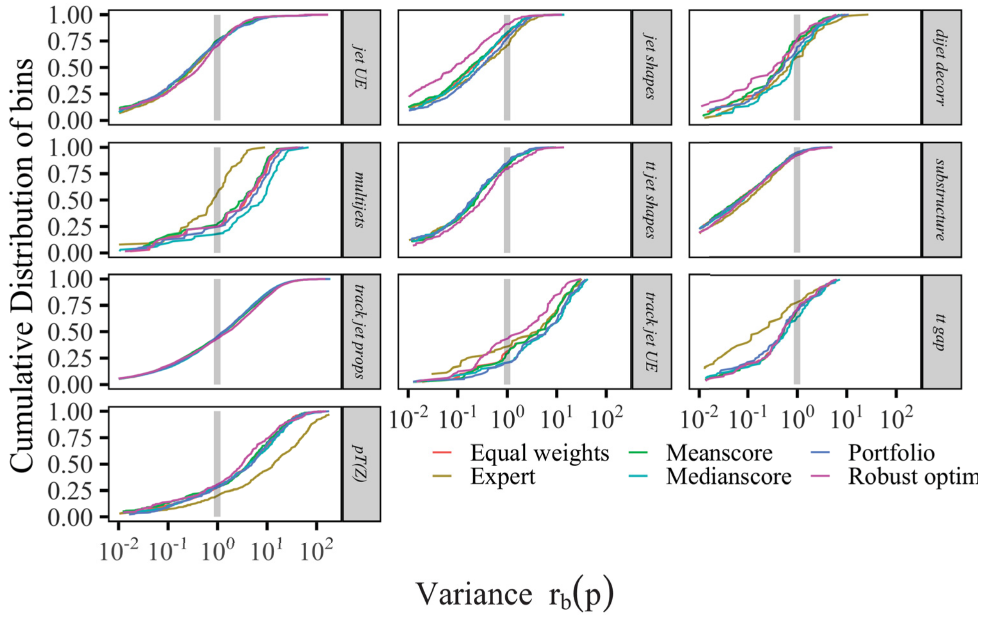


Figure 2: Cumulative distribution of bins (y axis) in each category of the A14 dataset at different bands of variance levels (x axis) given by $r_b(\mathbf{p}) = \frac{(f_b(\mathbf{p})-\mathcal{R}_b)^2}{\Delta f_b(\mathbf{p})^2+\Delta\mathcal{R}_b^2}$.

we look forward to hearing from you. To contribute, either raise an issue in the APPRENTICE project or contribute via code by forking the project. The APPRENTICE project is located at https://github.com/HEPonHPC/apprentice.

6 Acknowledgments

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, grant "HEP Data Analytics on HPC", No. 1013935. This work was also supported by the U.S. Department of Energy through grant DE-FG02-05ER25694, and by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research and Office of Nuclear Physics, SciDAC program through the FASTMath Institute under Contract No. DE-AC02-05CH11231 at Lawrence Berkeley National Laboratory.

References

[1] A. Buckley, H. Hoeth, H. Lacker, H. Schulz, J. von Seggern, The European Physical Journal C **65**, 331 (2010)

[2] R. Devore, X.M. Yu, Transactions of the American Mathematical Society **293**, 161 (1986)
[3] D.J. Newman, Michigan Math. J. **11**, 11 (1964)
[4] P. Gonnet, R. Pachón, L.N. Trefethen, Electron. Trans. Numer. Anal. **38**, 146 (2011)
[5] R. Pachón, P. Gonnet, J. Van Deun, SIAM J. Numer. Anal. **50**, 1713 (2012)
[6] J. Nocedal, S. Wright, *Numerical optimization* (Springer Science & Business Media, 2006)
[7] J.R. Shewchuk et al., *An introduction to the conjugate gradient method without the agonizing pain* (1994)
[8] R.H. Byrd, P. Lu, J. Nocedal, C. Zhu, SIAM Journal on scientific computing **16**, 1190 (1995)
[9] A.R. Conn, N.I. Gould, P.L. Toint, *Trust region methods* (SIAM, 2000)
[10] M. Powell, *Recent Research at Cambridge on Radial Basis Functions* (New Developments in Approximation Theory, pp. 215-232. Birkhäuser, Basel, 1999)
[11] *MPI for python*, https://mpi4py.readthedocs.io/en/stable/tutorial.html
[12] W.E. Hart, C.D. Laird, J.P. Watson, D.L. Woodruff, G.A. Hackebeil, B.L. Nicholson, J.D. Siirola, *Pyomo–optimization modeling in python*, Vol. 67, 2nd edn. (Springer Science & Business Media, 2017)
[13] W.E. Hart, J.P. Watson, D.L. Woodruff, Mathematical Programming Computation **3**, 219 (2011)
[14] *ipopt - a cython wrapper for the ipopt optimization solver.*, https://pythonhosted.org/ipopt/z
[15] M. Tanabashi, K. Hagiwara, K. Hikasa, K. Nakamura, Y. Sumino, F. Takahashi, J. Tanaka, K. Agashe, G. Aielli, C. Amsler et al. (Particle Data Group), Phys. Rev. D **98**, 030001 (2018)
[16] R.J. Barlow, C. Beeston, Comput. Phys. Commun. **77**, 219 (1993)
[17] M. Hoferichter, P. Klos, J. Menéndez, A. Schwenk, Phys. Rev. **D94**, 063505 (2016), `1605.08043`
[18] D.G. Cerdeño, A. Cheek, E. Reid, H. Schulz, JCAP **1808**, 011 (2018), `1802.03174`
[19] F. Feroz, M.P. Hobson, M. Bridges, Mon. Not. Roy. Astron. Soc. **398**, 1601 (2009), `0809.3437`
[20] F. Feroz, M.P. Hobson, E. Cameron, A.N. Pettitt, Instrumentation and Methods for Astrophysics (2013), `1306.2144`
[21] J. Buchner, A. Georgakakis, K. Nandra, L. Hsu, C. Rangel, M. Brightman, A. Merloni, M. Salvato, J. Donley, D. Kocevski, aap **564**, A125 (2014), `1402.0004`
[22] A. Fowlie, M.H. Bardsley, Eur. Phys. J. Plus **131**, 391 (2016), `1603.00555`
[23] ATLAS Collaboration (ATLAS), Tech. Rep. ATL-PHYS-PUB-2014-021, CERN, Geneva (2014), `https://cds.cern.ch/record/1966419`
[24] A. Buckley, *ATLAS PYTHIA 8 tunes to 7 TeV data*, in *Proceedings of the Sixth International Workshop on Multiple Partonic Interactions at the Large Hadron Collider*, CERN (CERN, Geneva, 2014), p. 29